基于 IFWA-ABC 的云计算资源任务调度算法的研究 *

陈　暄[1]，王大伟[1]，王常亮[1]，龙　丹[2]

(1. 浙江工业职业技术学院，浙江 绍兴 312000; 2. 浙江大学，杭州 310058)

摘　要：针对云计算资源任务调度效率低，资源分配不均的情况，将改进的烟花算法(improve fireworks algorithm)和人工蜂群算法(artificial bee colony)算法进行融合为 IFWA-ABC。首先，对云计算资源任务调度进行描述；其次，在 FWA 初始化中采用混沌反向学习和柯西分布进行优化，对核心烟花和非核心烟花的半径分别进行优化，将 FWA 中最优个体通过改进的 ABC 算法进行获得；最后，将 IFWA-ABC 算法用于云计算任务调度。仿真实验中，通过与 FWA、ABC 在虚拟机、执行时间、消耗成本、能量消耗指标对比中，IFWA-ABC 具有明显的优势能够有效地提高云计算资源分配效率。

关键词：烟花算法；人工蜂群算法；云计算；混沌反向学习

中图分类号：TP301.6　　doi: 10.3969/j.issn.1001-3695.2018.04.0213

Cloud computing task scheduling algorithm based on IFWA-ABC

Chen Xuan[1], Wang Dawei[1], Wang Changliang[1], Long Dan[2]

(1. *Zhejiang Industry Polytechnic College*, *Shaoxing Zhejiang* 312000, *China*; 2. *Zhejiang University*, *Hangzhou* 310058, *China*)

Abstract: Due to the low efficiency of cloud computing resource task scheduling and uneven resource allocation, this paper combined the improved fireworks algorithm and artificial bee colony algorithm into IFWA-ABC. Firstly, this paper described the cloud computing resource task scheduling. Secondly, it used the chaotic reverse learning and Cauchy distribution to optimize the FWA initialization. The radii of the core fireworks and the non-core fireworks are optimized respectively. The optimal individuals in the FWA are obtained by improving the ABC algorithm. Finally, the IFWA-ABC algorithm is used for cloud computing task scheduling. In the simulation experiment, IFWA-ABC has obvious advantages compared with FWA and ABC in terms of virtual machine, execution time, consumption cost and Energy consumption index, which can effectively improve cloud computing resource allocation efficiency.

Key words: fireworks algorithm; artificial bee colony algorithm; cloud computing; chaotic inverse learning

0　引言

云计算中的任务资源调度是衡量云计算中的一个重要组成部分,如何能够有效的、合理的分配任务资源之间的关系是云计算中效率能否提高的关键[1]。将智能算法引入到云计算中的资源调度已经成为了目前研究的主流,国内外学者一方面采用单一智能算法用于云计算资源调度,如 PSO 算法[2,3]、遗传算法[4,5]、蚁群算法[6,7]、人工蜂群算法[8];DAG 调度算法[9],烟花算法[10].另一方面将融合后的智能算法用于云计算资源调度。文献[11]提出了将粒子群算法和蚁群算法进行融合用于云计算资源调度,该算法优点提高了算法精度,提高了调度效率,缺点是缺乏与其他的更多智能算法进行对比的效果;文献[12]提出采用遗传算子生成初始信息素分布,通过双向收敛蚁群算子求出精确解,该算法优点是提高了求解精度和收敛速度,缺点是采用遗传算子的初始化信息素加强了算法的复杂性,提高了算法的求解时间;文献[13]提出对蚁群算法和粒子群算法分别进行改进,利用两种算法自身优势相结合的方式建立一种蚁群粒子群算法,用于云计算资源调度效率,该算法的优点是降低了消耗时间,缺点是降低了算法的精度。

本文以上研究的基础上,针对 FWA 存在的问题,对种群进行初始化、烟花半径进行改进,并在选择策略中采用 ABC 进行改进,融合后的 IFWA-ABC 算法用于云计算资源调度。仿真实验中在负载均衡、执行时间和消耗成本方面与 FWA,ABC 算法相比具有一定的优越性。

收稿日期：2018-04-02；**修回日期：**2018-05-21　　**基金项目：**国家自然科学基金资助项目（LQ18A010003，11426205）；绍兴市科技局资助项目（2015B70013）

作者简介：陈暄（1979-），男，江西人，副教授，硕士，主要研究方向为云计算、无线传感（chenxuan1979@sina.com）；王大伟（1982-），男，安徽人，讲师，硕士，主要研究方向为云计算；王常亮（1973-），男，山西人，讲师，主要研究方向为云计算，算法设计；龙丹（1975-）男，湖南人，讲师，博士，主要研究方向为图像处理和分析、算法设计.

1　云计算资源调度任务简述

云计算中的任务调度效率是衡量云计算资源调度效率高低的重要指标，本文从云计算资源任务调度分的虚拟机负载、执行时间、花费成本三个方面进行描述，并用此作为云计算资源任务调度的效果评价依据

1.1　虚拟机负载

虚拟机负载是云计算任务资源中调度中的重要组成部分，由于云计算中的资源会随着任务进行动态的变化,因此,记录单个虚拟机的相关参数无法反馈整个云计算动态情况。本文采用如下的方式记录虚拟机:

$$ResLoad(VM_j) = [id_j, VM_j, lastTime_j] \tag{1}$$

其中：id_j 表示具有唯一性的虚拟机的编号; VM_j 表示虚拟机自身的性能参数; $lastTime_j$ 表示最后一个任务在虚拟机中的执行时间。显然,当每一个任务 T_i 分配给虚拟机 VM_j 的时候,最终任务执行时间 $lastTime_j$ 需要进行修改:

$$lastTime_j = lastTime_j^{'} + taskTime_{ij} \tag{2}$$

$$taskTime_{ij} = \frac{T_{i_taskLtenth}}{VM_{j_cpu}} + \frac{T_{i_InputFilesize}}{VM_{j_bw}} + t_{wait} \tag{3}$$

其中：$lastTime_j^{'}$ 表示到上一次使用 VM_j 所完成时间; $taskTime_{ij}$ 表示将任务 T_i 分配到虚拟机 j 上所需要的执行时间如式(3)所示, $T_{i_taskLtenth}$ 和 $T_{i_InputFilesize}$ 对应任务 T_i 中的任务长度和相关的输入信息, VM_{j_cpu} 和 VM_{j_bw} 分别表示虚拟机的计算能力和通信带宽。

1.2　执行时间

云计算中的时间花费是衡量云计算资源调度效果的重要组成部分,本文采用 $vmTime(VM_j)$ 来表示所有任务的执行,也就是说每个虚拟机实际完成时间就是分配到该虚拟机上的所有任务的时间总和, $\sum_{i \in VM_j} taskTime_{ij}$ 表示任务分配到第 j 个虚拟机 VM_j 上的预测执行时间如式(4)表示,因此一个任务 I 的执行时间如式(5)所示。

$$vmTime(VM_j) = \sum_{i \in VM_j} taskTime_{ij} \tag{4}$$

$$ResTime(I) = \sum_{j=1}^{m} vmTime(VM_j) \times VM_{j_cost} \tag{5}$$

1.3　成本约束

云计算资源任务调度中除了时间之外,还有另外一个重要的衡量因素-任务成本消耗,它是云计算资源任务调度的重要组成部分。每一个任务的总的成本如下：

$$ResCost(I) = \frac{finishCost(I) - finishCost_{min}}{finishCost_{max} - finishCost_{min}} \tag{6}$$

其中：$finishCost_{max}$ 和 $finishCost_{min}$ 分别代表任务预测中的最大成本和最小成本,也就是部署在虚拟机上的最大时间和最小时间对应的费用,具体计算公式如下:

$$finishCost_{max} = finishTime_{max} \times Max(VM_{j_cost}) \tag{7}$$

$$finishCost_{min} = finishTime_{min} \times Min(VM_{j_cost}) \tag{8}$$

1.4　能量消耗

云计算任务能量消耗关系到云计算资源调度的效果,本文采用 $vmEnger(VM_j)$ 来表示所有任务的能量消耗,也就是说每个虚拟机实际能量消耗就是分配到该虚拟机上的所有任务的能量消耗的总和, $\sum_{i \in VM_j} taskenger_{ij}$ 表示任务分配到第 j 个虚拟机 VM_j 上的能量消耗如式(9)表示,一个任务的能量消耗如式(10)所示。

$$vmEnger(VM_j) = \sum_{i \in VM_j} taskenger_{ij} \tag{9}$$

$$ResEnger(I) = \sum_{j=1}^{m} vmEnger(VM_j) \times VM_{j_cost} \tag{10}$$

因此,衡量任务 I 对应的资源调度的效果的函数如下:

$$F(I) = w \times ResLoadVM(I) + t \times ResTime(I) + c \times ResCost(I) + e \times ResEnger(I) \tag{11}$$

其中：w , t , c 和 e 分别为虚拟机,时间,成本及能量消耗的影响因子,并且都是 0~1 的随机数,且 $w+t+c+e=1$ 。显然,可以通过调节 w , t , c 和 e 四个因子的值来设定虚拟机负载、执行时间,消耗成本及能量消耗在任务资源选择过程中的影响权重,这样既满足了系统对任务执行时间的约束,又满足了用户降低成本的要求。

2　FWA 算法

Tan 和 Zhu 根据烟花爆炸的形态于 2010 年提出了烟花算法(fireworks algorithm, FWA) [14]，它具有随机性、局部性和多样性的特点，其主要用于问题的优化。FWA 中的每一个烟花作为优化问题中的一个可行解，根据适应度函数计算每一个烟花产生火花数值，适应度值越好的烟花产生对应的火花数目越多，反之，则越少，并且烟花爆炸过程中变异过程保证种群的多样性。FWA 主要包括爆炸算子、变异算子和选择策略三个部分组成。

2.1　爆炸算子

在算法的初始化中，需要对产生的烟花的位置对应的适应度值进行初步的评估，适应度值较好的烟花（一般被成为核心烟花）能够获得更多的资源，并且能够在小范围内产生更多的烟花，具有较强的局部搜索能力，而适应度值较差的烟花（一般成为非核心烟花）只能获得较少的火花，但是具有一定的全局搜索能力。因此，每个烟花的爆炸半径和所产生的火花数目根据种群中的其他烟花的适应度值计算得到。即爆炸半径 A_i 和爆炸火花数目 S_i 的计算如下：

$$A_i = \hat{A} * \frac{f(x_i) - Y_{min} + \varepsilon}{\sum_{i=1}^{N}(f(x_i) - Y_{min}) + \varepsilon} \tag{12}$$

$$S_i = M \times \frac{y_{\max} - f(x_i) + \varepsilon}{\sum_{i=1}^{N}(y_{\max} - f(x_i)) + \varepsilon} \tag{13}$$

其中：$y_{\min} = \min(f(x_i))$ 和 $y_{\max} = \max(f(x_i))$ 分别表示当前种群中的适应度值最小值和最大值。使用 $\hat{A}$ 来调整爆炸半径，M 是一个常数，用来设定火花产生数目的大小，ε 为一个机器最小参数用来避免零操作。为了降低位置值好的烟花产生数量较少的爆炸火花，需要对火花个数进行限制：

$$\hat{S}_i = \begin{cases} round(a*M), S_i < aM \\ round(b*M), S_i > bM, a < b < 1 \\ round(S_i), otherwise \end{cases} \tag{14}$$

其中：a,b 为常数，$round(\bullet)$ 为四舍五入的取整函数。

2.2 变异算子

为了能够增加种群的多样性,使用变异算子来产生变异火花。这种变异火花即高斯变异火花,主要是在种群中随机选择一个烟花个体,并在一定的维度上进行高斯变异操作得到公式如式(15)所示。

$$\hat{x}_{ik} = x_{ik} \times e \tag{15}$$

其中，e 为 $N(1,1)$ 的高斯变异操作。

2.3 选择策略

为了能够在当前的种群选择有效的个体传递到下一代中，在经历过以上 2 个操作之后，算法选择一定数量的个体作为下一代的烟花，其选择的概率为

$$p(x_i) = \frac{R(x_i)}{\sum_{x_j \in K} x_j} \tag{16}$$

$$R(x_i) = \sum_{x_j \in K} d(x_i - x_j) = \sum_{x_j \in K} \| x_i - x_j \| \tag{17}$$

其中：$R(x_i)$ 为当前个体候选者集合除了 x_i 之外所有个体的距离之和。通过以上的方式操作方式后，如果个体密度较高，该个体周围被选择的概率会降低。

3 基于 IFWA-ABC 的云计算资源调度算法

针对 FWA 算法自身过早收敛,解精度低,容易陷入局部最优的问题,本节对 FWA 算法在初始化,烟花半径和选择策略方面进行改进,将得到的 IFWA-ABC 算法并用于云计算资源调度

3.1 改进的 FWA 算法

3.1.1 混沌反向学习和柯西分布优化初始位置

FWA 算法的初始解通常采用随机方式，这种随机处理方法在一定程度上对最优解的产生具有一定的影响，假设两者极端的情况，一种是产生的随机解距离最优解很近的时候，算法的收敛速度就会很快，另一种是产生的随机解距离很远或者反向的时候，导致算法需要消耗更长的时间才能够完成收敛。针对这种问题，本文引入了混沌反向学习概念来初始化解，即在当前解的同时求得反向解，通过对两者的对比，从中选择较好的一个解，这样可以在很大程度上提高算法的执行效率。根据 FWA 算法中的烟花个体的位置，给出 Logistic 映射产生混沌状态下的初始值，如式(18)所示，计算其位置的反向解如式(19)所示。式中，$x_{\max,j}$ 和 $x_{\min,j}$ 是搜索空间中的候选解的最大值和最小值

$$x_{i,j} = x_{\min,j} + \delta_{i,j} \bullet (x_{\max,j} - x_{\min,j}) \tag{18}$$

$$x^*_{i,j} = x_{\min,j} + x_{\max,j} - x_{i,j} \tag{19}$$

因此，将反向后的解与原来解组成一个群体，按照适应度值的由高到低进行排列，选择前一半的值作为种群的初始化的解。

在标准的 FWA 算法中，新一代的烟花个体散落在父代个体周围，产生的解会随着迭代次数的增加而不断的减小，将搜索的范围逐渐搜索到父代个体附近，这样使得算法容易陷入局部最优，而降低求出全局最优解的效率，采用柯西分布生成的随机数能够使得搜索区域更加广泛，因此能够提高算法的全局搜索能力，因此采用如下的密度函数进行计算。

$$\begin{aligned} & f(x,\gamma) = \frac{1}{\pi}\left[\frac{\gamma}{x^2+\gamma^2}\right] \\ & s.t.\ \gamma_{iter} = \frac{(iter_{\max} - iter)^n}{(iter_{\max})^n}(\gamma_{initial} - \gamma_{final}) + \gamma_{final} \end{aligned} \tag{20}$$

其中：$iter$ 表示迭代次数，$iter_{\max}$ 表示最大迭代次数，$\gamma_{initial}$ 和 γ_{final} 分别表示柯西分布中的尺度参数的初始值和最大值。

3.1.2 设定阈值优化非核心烟花半径

根据 FWA 算法得知，当烟花在爆炸的时候，会在它的周围一定的区域中产生大量的火花，火花的范围就是烟花的半径，核心烟花是烟花种群中的适应度值最小的烟花个体，而非核心烟花半径如公式(12)所示，很显然这种方式会发现适应度值最小的烟花的半径非常小，几乎接近 0，因此浪费了资源。为了避免这种情况的发生，设定最小半径阈值，其表现形式如下：

$$A_{i,k} = \begin{cases} A_{\min,k}, if\ A_{i,k} < A_{\min,k} \\ A_{i,k}, otherwise \end{cases} \tag{21}$$

阈值 $A_{\min,k}$ 的设定并不是一个固定的值，这是因为算法在迭代过程中会产生的动态的效果，因此采用非线性递减方式。

$$A_{j,k}(t) = (A_{start} - A_{end} - d_{\max})e^{\frac{t}{1+d_{\max}}} \tag{22}$$

其中：$d_{\max}$ 为最大函数评估次数，t 为当前函数评估次数，A_{start} 和 A_{end} 分别表示最初的爆炸半径和最终半径数值。

3.1.3 权重算子优化核心烟花半径

文献[15]中提到需要对核心烟花的爆炸半径进行一定的调整才能使得算法的性能得到改进，显然这种改进存在的一定问题，其主要表现在来自局部搜索能力的提高，烟花个体之间相互并没有交流，提出的核心烟花半径计算过于简单，仅仅考虑缩放而直接忽略了搜索过程，容易造成算法陷入局部最优。使得算法过早的收敛，导致算法难以跳出多峰值的时候局部最优解的束缚，本文在此基础上采用粒子群算法中的权重算子，其主要目的是利用当前最好烟花位置和历史最优信息，从而避免了陷入局部最优。表达如下：

$$\hat{X}_i = X_{best} + c*(X_{best} - X_i) \tag{23}$$

$$c = \begin{cases} randc(0.1,0.5) & if\ rand < \dfrac{t_1}{t_1 + t_2} \\ randc(0.5,1.0) & otherwise \end{cases} \tag{24}$$

其中：X_{best} 表示当前位置中最好的烟花位置，c 为权重因子控制先产生的火花与当前最好烟花位置之间的距离，使得烟花个体具有向种群中最优秀的个体学习的能力从而向群体内历史最优点靠近。$randc(x,y)$ 是位置参数 x 和尺度参数为 y 的柯西分布，当位置参数小的时候，有利于进行局部搜索，使用 t_1 表示位置参数小的时候柯西分布，当位置参数大的时候，有利于进行全局搜索，使用 t_2 表示参数大的柯西分布。

3.1.4 基于 ABC 算法选择策略

在 FWA 算法中，每一次迭代过程中优秀烟花个体都能传递到下一代种群中，FWA 算法主要是选择适应度值最小的个体，而剩余个体都采用随机状态，显然，这种挑选的策略存在一定的弊端，虽然在某次迭代中，按照适应度值小的选择个体，但从全局来看不一定能够保证选中的个体就一定比剩余随机状态的个体要好，同时剩余的个体采用随机的状态，从整体上不利于算法全局最优解的诞生，针对这个情况，本文引入人工蜂群算法(artificial bee colony,ABC 算法)，ABC 算法具有简单易实现、控制参数少、鲁棒性强以及全局搜索能力较强等特点，通过每个人工蜂个体优化作用最终找到最优的个体。在 ABC 算法中，分为 3 种个体，分别是雇佣蜂、观察蜂和侦察蜂。每一个雇佣蜂对应一个确定的蜜源(FWA 中个体的位置)，并在迭代过程中对蜜源的周围位置进行更新，根据蜜源的适应度值采用轮盘的方式雇佣观察蜂进行采蜜，以便获得新的蜜源，当一定搜索次数之后没有新蜜源出现就放弃该蜜源，雇佣蜂变为侦察蜂随机搜索的新蜜源。

a) 初始化。统计 FWA 中的种群中所有个体 SN，并计算每个个体的适应度值，产生可行解为

$$x_{ij} = x_{\min,j} + rand(0,1)(x_{\max,j} - x_{\min,j}) \tag{25}$$

其中：x_i（$i = 1,2,\cdots SN$）为 D 维向量，$j \in \{1,2,\cdots D\}$

b) 蜜源最优值。在当前的蜜源相关领域中进行搜索，借助 DE 算法的思想，因此新的蜜源公式为

$$v'_{i,j} = x_{r1,j} + \varphi_{i,j}(x_{i,j} - x_{r2,j}) + \phi_{i,j}(x_{r3,j} - x_{r4,j}) \tag{26}$$

$$v''_{i,j} = x_{best,j} + \varphi_{i,j}(x_{i,j} - x_{r1,j}) + \phi_{i,j}(x_{r2,j} - x_{r3,j}) \tag{27}$$

$$v_{i,j} = p \bullet v'_{i,j} + (1-p) \bullet v''_{i,j} \tag{28}$$

其中：$j \in \{1,2,\cdots D\}$ 为随机选择的目标，$i \in \{1,2,\cdots SN\}$，φ_{ij} 和 $\phi_{i,j}$ 分别为[-1, 1]和[0.5, 1]的随机数，p 都是随机选择的整数，$x_{best,j}$ 是当前全局最优解，显然，借助差分算法的影响式(26)能够有效地保持种群的多样性，式(27)可以提高收敛速度，增强算法的开发能力，为了能够将两者进行有效的结果，采用式(28)来控制新的蜜源位置。

c) 雇佣蜂概率。观察蜂选择雇佣蜂概率如式(29)所示。

$$P_i = \frac{fit(x_i)}{\sum_{n=1}^{SN} fit(x_n)} \tag{29}$$

其中：$fit(x_i)$ 表示第 i 个解的适应度值对蜜源的丰富程度，当蜜源越丰富，被观察蜂选择的概率就越大。

在 ABC 算法中，当 FWA 中个体位置对应的蜜源在迭代过程中没有进行改进，就放弃该蜜源即该烟花个体，并且将该蜜源对应的烟花个体位置放到禁忌表中，同时将该蜜源对应的雇佣蜂转为侦察蜂，并根据式(16)随机产生一个 FWA 的个体位置代替原有的个体。

3.2 基于 FWA-ABC 算法的云计算资源调度

根据融合后的两种算法的优点，结合云计算中资源任务调度步骤如下，流程如图 1 所示。

a)将云计算任务调度的方案与 FWA 算法中的烟花位置进行一一对应，找到最佳的烟花位置即为最佳的任务调度方案；

b)初始化 FWA 算法的初始化参数，ABC 算法的参数，设定迭代次数；

c)按照式(18)~(20)对 FWA 算法种群进行初始化；

d)对烟花半径进行分别优化，核心烟花半径按照式(21)(22)进行优化，对非核心烟花半径按照式(23)(24)进行优化；

e)将烟花个体通过 ABC 算法中的公式(25)-(29)进行处理；

f)当达到最大迭代次数的时候，算法结束，转步骤 g)，否则转步骤 c)；

g)得到最优的烟花位置，即得到最优的云计算任务调度方案。

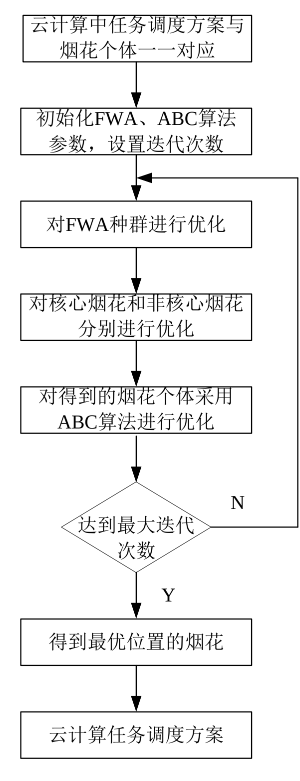


图 1　调度流程

4 实验仿真

4.1 算法性能

为了说明本文算法的性能，将 IFWA-ABC 算法与 FWA 算法、ABC 算法在 3 个经典测试函数中进行对比。每一种算法运行 200 次，比较结果如表 1 所示

$$f_1(x)=\sum_{1}^{n}x^2 \tag{30}$$

$$f_2(x)=\sum_{i=1}^{n}\left(\sum_{j=1}^{i}x_j\right)^2 \tag{31}$$

$$f_3(x)=\sum_{i=1}^{n}(x_i^2-10\cos(2\pi_i)+10) \tag{32}$$

表 1 测试结果对比

维数	算法	最优值	最差值
	IFWA-ABC	3.78214e-2	5.38272e-2
f_1	FWA	4.12712e-2	7.36234e-2
	ABC	5.89234e-2	9.39802e-2
	IFWA-ABC	0.0023682	0.0873232
f_2	FWA	0.0063982	0.1980327
	ABC	0.0298217	0.2739231
	IFWA-ABC	0.5392873	2.9837123
f_3	FWA	1.7309431	4.2938731
	ABC	4.983491	6.3940211

从表 1 中发现 IFWA-ABC 的性能相比 FWA 算法具有显著的提高，说明本文的算法效果较好。

4.2 云计算任务调度

为了说明 IFWA-ABC 算法在云计算中资源调度的效果，将 IFWA-ABC 算法与 FWA、ABC 算法在虚拟机负载均衡、消耗时间和成本花费进行对比。硬件环境选 CPU 为酷睿 i3，内存为 4GDDR，硬盘容量为 1000G，软件环境为 Linux 操作系统，采用 Cloudsim 模拟云计算环境。选取三种不同的任务集合，分别为 Task1[100,1000]，Task2[1000,10000] 和 Task3[10000,100000]，资源数目为 500。FWA 算法，ABC 算法的相关参数以文献[8]，文献[10]为主，IFWA-ABC 算法中的相关参数设置如下：$\gamma_{initial}$ 和 γ_{final} 分别 10 和 50，$d_{\max}$ 为 100，A_{start} 和 A_{end} 分别为 2.5 和 10，φ_{ij} 和 $\phi_{i,j}$ 分别为 0.5 和 1，p 为 0.5。

4.3 负载均衡对比

本文将三种任务集合在 10 个资源点中 {s1,s2,s3,s4,s5,s6,s7,s8,s9,s10}，s1 到 s10 处理任务为分别为 50,100,150,200,250,300,350,400,450 和 500，三种任务集合状态下的三种算法的对比效果如图 2-4 所示，每个云计算的资源处理点的能力不同，因此负载都不相同，从图 2-4 中发现，当任务数量较小的时候，三种算法的负载均衡都差不多，当任务的数量逐渐增大的时候，IFWA-ABC 算法的负载均衡，而 FWA 和 ABC 算法的负载产生的数值不同，这说明 IFWA-ABC 算法在云计算资源分配效果明显，负载稳定。

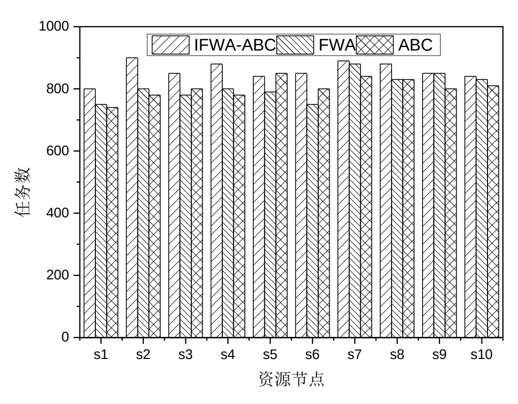


图 2 三种算法 Task1 中的负载均衡对比

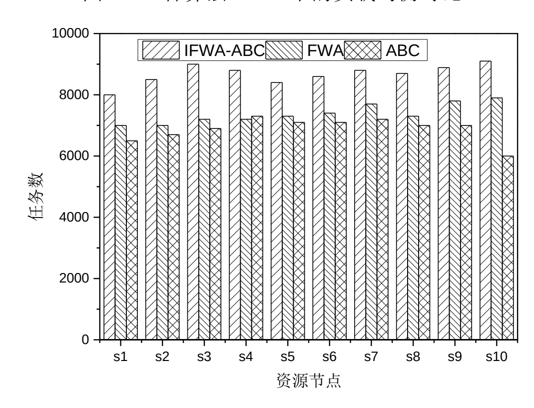


图 3 三种算法 Task2 中的负载均衡对比

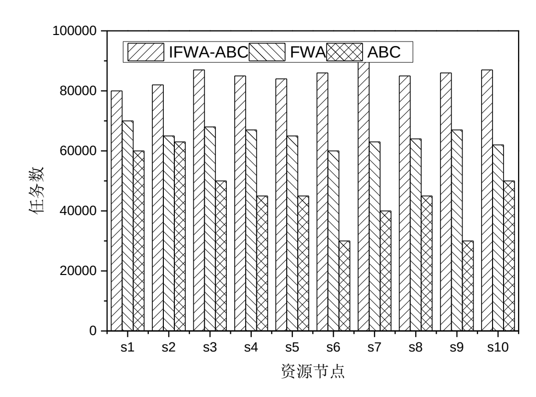


图 4 三种算法 Task3 中的负载均衡对比

4.4 执行时间对比

图 5~7 显示了三种任务集合下的执行时间对比情况，从图中可以发现，三种算法的执行时间都随着任务数量的增加具有不同程度的增加，当任务数量较小的时候，三种算法的执行时间相差不大，当任务数量较大的时候，三种算法执行时间之间的相差明显，IFWA-ABC 算法的执行时间明显优于 FWA、ABC 算法，这说明 IFWA-ABC 算法能够适应云计算下任务资源调度。

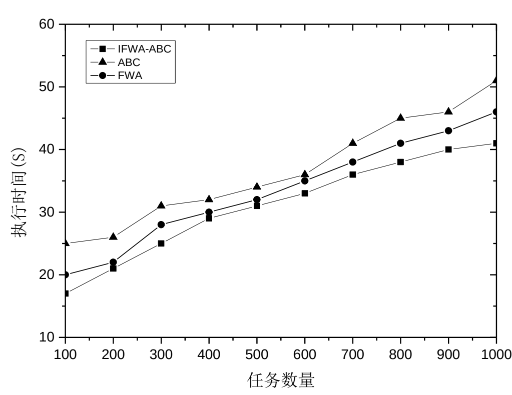


图 5 三种算法 Task1 中的执行时间对比

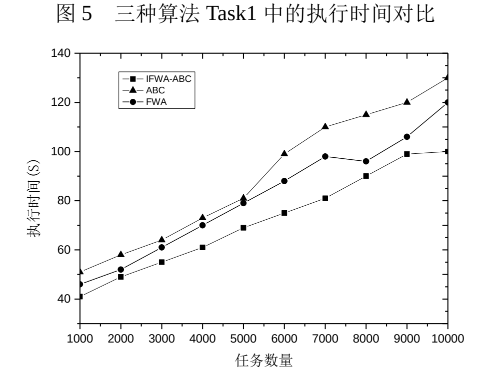


图 6 三种算法 Task2 中的执行时间对比

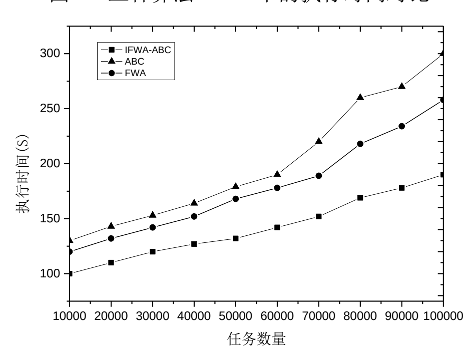


图 7 三种算法 Task3 中的执行时间对比

4.5 消耗成本对比

图 8~10 显示了三种任务集合下的消耗成本的对比情况,从图中可以发现,三种算法的消耗成本都伴随着任务数量的增多而变大,当任务数量较小的时候,三种算法消耗成本的曲线都在平缓的增长,当任务数量逐渐增大的时候,三种算法的消耗成本的曲线都在不同程度的增加,且 IFWA-ABC 的曲线增长幅度平缓,而 FWA 和 ABC 算法幅度较大,这说明 IFWA-ABC 相比于 FWA,ABC 算法在消耗成本方面具有较大的优势,能够适应。

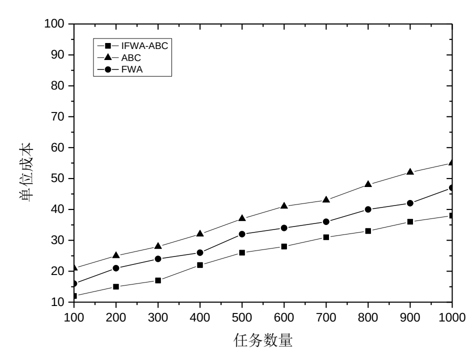


图 8 三种算法 Task1 中的消耗成本对比

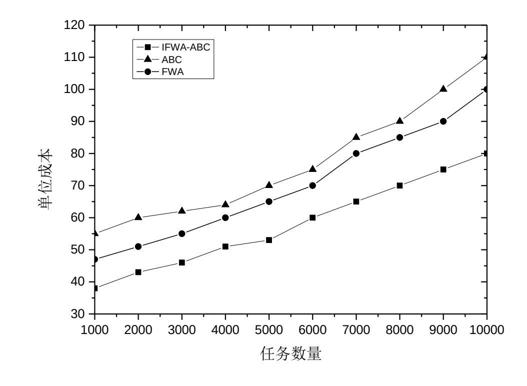


图 9 三种算法 Task2 中的消耗成本对比

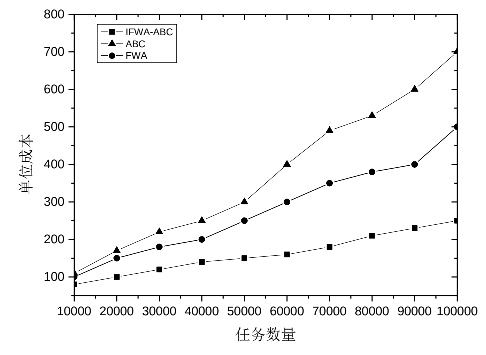


图 10 三种算法 Task3 中的消耗成本对比

4.6 能量消耗对比

图 11~13 显示了三种任务集合下的能量消耗对比的结果,在任务数量较小的时候,相互之间的能量消耗差别并不是很大,但总体上 IFWA-ABC 算法占据一定的优势,当任务数量逐渐增大的时候,IFWA-ABC 相比于 FWA 和 ABC 算法具有明显的优势,这说明了 IFWA-ABC 算法能够有效的降低能量消耗

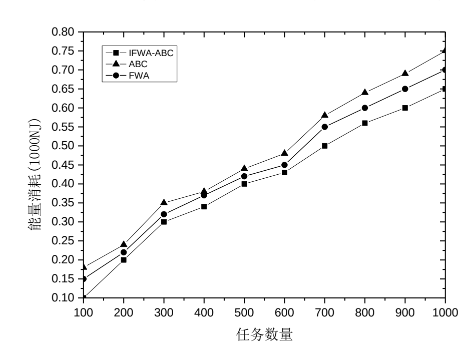


图 11 三种算法在 Task1 中的能量消耗对比

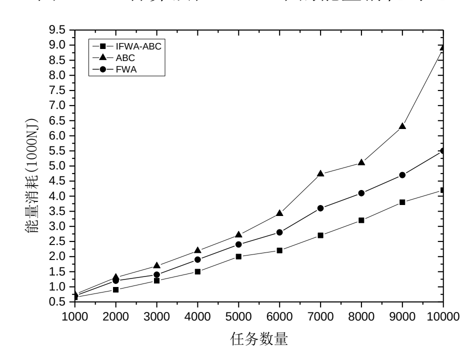


图 12 三种算法在 Task2 中的能量消耗对比

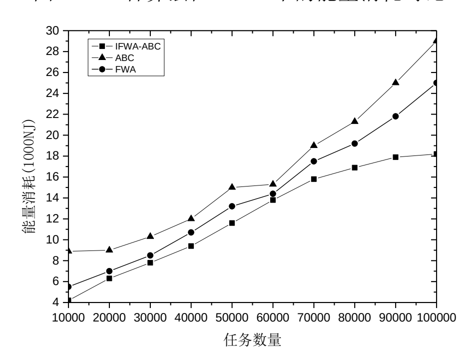


图 13 三种算法在 Task3 种的能量消耗对比

5 结束语

针对云计算任务资源的问题,本文将 FWA 算法用于任务调度,并对 FWA 算法自身进行了改进,将 ABC 算法与之进行融合

形成 IFWA-ABC,,仿真实验从均衡负载、执行时间、消耗成本和能量消耗等方面说明了 IFWA-ABC 算法具有良好的效果,能够适应云计算下任务调度。

参考文献：

[1] 张建勋，古志民，郑超．云计算研究进展综述 [J]. 计算机应用研究, 2010, 27 (2): 429-433. (Zhang Jianxun, Gu Zhimin, Zheng Chao. Survey of research progress on cloud computing [J]. Application Research of Computers, 2010, 27 (2): 429-433.)

[2] Masdari M, Salehi F, Jalal M, *et al.* A survey of pso-based scheduling algorithms in cloud computing [J]. Journal of Network and Systems Management, 2017, Vol. 25, No. 1, pp: 122-158

[3] Kumar N, Patel P. Resource management using ANN-PSO techniques in cloud environment [C]// Proc of International Congress on Information and Commuication Technology. 2016: 419-428

[4] Shahdi-Pashaki S, Teymourian E, Tavakkkoli Moghaddam R. New approach based on group technology for the consolidation problem in cloud computing-mathematical model and genetic algorithm [J]. Computational and Applied Mathematics, 2016, DOI: 10. 1007/s40314-016-0362-4

[5] Aziza H, Krichen S. Bi-objective decision support system for task-scheduling based on genetic algorithm in cloud computing [J]. Computing, 2018, 100 (2): 65-91.

[6] Ragmani A, Omri A E, Abghour N, *et al.* A performed load balancing algorithm for public Cloud computing using ant colony optimization [C]// Proc of the 2nd International Conference on Cloud Computing Tehnologies and Application. 2016: 7847703

[7] Satapathy. A basic simulation of ACO algorithm under cloud computing for fault tolerant [C]// Proc of International Conference on Data Engineering and Communication Technology. 2017: 465-472.

[8] 杨海军．云计算环境下人工蜂群作业调度算法设计 [J]. 数学的实践与认识，2012, 42 (10): 115-120. (Yang Haijun. A job scheduling algorithm based on artificial bee colony in cloud computing [J]. athematics in Practice and Theory, 2012, 42 (10): 115-120.)

[9] 徐健锐，朱会娟．云计算环境中面向 DAG 任务的多目标调度算法 [J/OL]. 计算机应用研究，2019, 36 (1) . [2018-01-10]. http://www. arocmag. com/article/02-2019-01-023. html. (Xu Jianrui, Zhu Huijuan. Multi-objective scheduling algorithm of DAG tasks in cloud computing [J]. pplication Research of Computer, 2019, 36 (1) . [2018-01-10]. http://www. arocmag. com/article/02-2019-01-023. html.)

[10] 黄伟建，郭芳．基于烟花算法的云计算多目标任务调度 [J]. 计算机应用研究，2017, 34 (6): 1718-1720. (Huang Jinwei, Guo Fang. Multi-objective task scheduling based on fireworks algorithm in cloud computing [J]. pplication Research of Computers, 2017, 34 (6): 1718-1720.)

[11] Chen X. Task Scheduling of cloud computing using integrated particle swarm algorithm and ant colony algorithm [J]. Cluster Computing, DOI: 10. 1007/s10586-017-1479-y, 2017

[12] 赵俊普，殷进勇，金同标，等．遗传蚁群算法在云计算资源调度中的应用 [J]. 计算机工程与设计, 2017 (3): 693-697. (Zhao Junpu, Yin Jinyong, Jin Tong biao, *et al.* Application of genetic ant colony algorithm in cloud computing resource sheduling [J]. Computer Engineering and Design, 2017, 38 (3): 693-697.)

[13] 萨日娜．基于蚁群粒子群优化算法的云计算资源调度方案 [J]. 吉林大学学报：理学版，2017, 55 (6): 1518-1522. (Sa Rina. Cloud computing resource scheduling scheme based on ant colony particle swarm optimization algorithm [J]. Journal of Jilin University: Sci Ed, 2017, 55 (6): 1518-1522.)

[14] Tan Y, Zhu Y. Fireworks algorithm for optimizaion [C]. Berlin: Springer, 2010: 355-364

[15] Zheng S, Janecek A, Li J, *et al.* Dynamic search in fireworks algorithm [C]// Evolutionary Computation. 2014: 3222-3229.